Army Research Laboratory

Enhancements to OneSAF Killer/Victim Scoreboard Capabilities

by Janet F. O'May

ARL-TR-3758 April 2006

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5067

ARL-TR-3758 **April 2006**

Enhancements to OneSAF Killer/Victim Scoreboard Capabilities

Janet F. O'May
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE					*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* April 2006	**2. REPORT TYPE** Progress	**3. DATES COVERED (From - To)** September 2004–September 2005
4. TITLE AND SUBTITLE Enhancements to OneSAF Killer/Victim Scoreboard Capabilities		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Janet F. O'May		**5d. PROJECT NUMBER** P622783
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRD-ARL-CI-CT Aberdeen Proving Ground, MD 21005-5067		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-3758
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The U.S. Army Research Laboratory uses the combat simulation One Semi-Automated Forces to support research in portraying battlespace areas of influence and determining critical course of action events through data mining. The combat simulation was originally modified to provide near real-time information on battle entities and direct-fire events. These data were obtained through the simulation but required some user interaction. This early work provided the framework for a killer/victim scoreboard capability. In an effort to expand this ability, the software has now been modified to collect required data automatically and to provide information on all indirect-fire events. These new capabilities further our ability to obtain reliable and complete data from the combat simulation in support of our ongoing projects.

15. SUBJECT TERMS

OneSAF, killer/victim scoreboard, indirect fire, simulation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Janet F. O'May
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UNCLASSIFIED	30	**19b. TELEPHONE NUMBER** (*Include area code*) 410-278-4998

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures **iv**

1. Introduction **1**

2. Background **1**

3. Methods/Procedures **3**

3.1 Data Collection Modifications 3

3.2 Indirect-Fire Modifications 4

4. Results and Discussion **5**

5. Conclusions/Recommendations **6**

6. References **7**

Appendix A. Changes in OneSAF's libsrc/libstatmon Directory **9**

Appendix B. Changes in OneSAF's libsrc/libentity Directory **13**

Appendix C. Changes in OneSAF's libsrc/libifdam Directory **15**

List of Symbols, Abbreviations, and Acronyms **23**

Distribution List **24**

List of Figures

Figure 1. Information for one entity from the data collection file. 2
Figure 2. Vehicle table sample....3
Figure 3. Direct-fire event....3
Figure 4. Sample from the indirect-fire file. 5

1. Introduction

The U.S. Army Research Laboratory (ARL) Battlespace Decision Support Team (BDST) uses combat simulations for its Battlespace Terrain Ownership (BTO) (*1*) and data mining projects (*2*). It is critical that accurate and timely data is obtained from simulations to support these efforts. We are currently using One Semi-Automated Force (OneSAF) Testbed Baseline (OTB)* version 2.0. The simulation was originally developed by the U.S. Army to support training. Over the years the simulation has expanded to support all domains.

BDST completed work in 2001 that provided a new capability to OneSAF. This capability was entitled the Killer/Victim Scoreboard (KVS). The work was originally developed for OTB version 1.0. When the Program Executive Office for Simulation, Training, and Instrumentation (PEO-STRI) placed a call for OTB v2.0 enhancements, the KVS code was submitted. The KVS is now inherent in OTB v2.0 and can be invoked with a configuration flag. However, KVS only tracked direct-fire results and needed to be expanded. In addition, the collection of information on battlespace entities required user intervention. New code has now been developed and tested that tracks indirect-fire events and allows for automatic data collection. This new capability furthers our efforts to obtain reliable and complete data from simulations to feed current projects.

2. Background

The traditional method for evaluating battle outcome is to determine if objectives were met and to examine force attrition. Part of BDST's objective was to come up with novel measures of effectiveness. To evaluate courses of actions, we wanted more detail from the battlespace. KVS gave us that level of detail. Implementing KVS was a two-fold process. The first step was to modify OneSAF to provide a listing of all entities in the battlespace and associated characteristics (*3*). This data collection capability originally occurred whenever the OneSAF code entered a certain software routine, and required user intervention to enable this functionality. Now, we have modified OneSAF again to collect this data without user intervention. The software collects data every 60 s by default, or a user can modify a file to collect data at any time period.

Figure 1 provides a sample of the information that is collected for each entity in the battlespace at either the default or user-specified time. Information is provided for each individual entity and their aggregation. If a company of main battle tanks is in the battlespace, information will be

*OneSAF software is sponsored by Program Executive Office for Simulation, Training, and Instrumentation (PEO-STRI).

```
Current Count 1 at Time: 1122577472
-----------------------------------
-----------------------------------
MARKING: 100A62
VEHICLE: vehicle_USSR_T80
VTAB:  1037
X = 10005.50  Y = 36515.00   Z = 938.11  CELL = 0
Vehicle           Authorized/Undamaged/Catastrophic/Firepower/Mobility
                                          Damage      Damage    Damage
T-80                        1          1           0           0          0

Equip/Supplies:           Current Lvl      Resupply Lvl      Avg Per Veh
Fuel (Fuel) (gallons)          499.39            499.95            499.39
125HEAT (125HEAT)               28.00             28.00             28.00
125SABOT (125SABOT)             12.00             12.00             12.00
D (D)                         2000.00           2000.00           2000.00
Songster (Songster)              4.00              4.00              4.00

Not aware of any vehicles.
```

Figure 1. Information for one entity from the data collection file.

provided for the company, the platoons in the company, and each vehicle. This example provides the time at which the data were collected, the entity type, the call sign, a unique identifier (vehicle table, or VTAB, entry), entity location, entity status, the current logistic levels (fuel and ammunition), and if the vehicle has spotted any other entities. The data are placed in a file created for each simulation run. The file is named with a unique timestamp, identified, and the suffix of *dc* for data collection. The unique timestamp is provided through a system call in the C language to the time function. This timestamp is used for all files created during the simulation execution. This allows all files relating to one simulation execution to be identified and correlated.

The next phase in the KVS process was to create two additional files. These files provide a short listing of every entity in the battlespace and direct-fire occurrence (*4*). The first file is the vehicle table file that lists all entities. The data include the unique identifier (VTAB four digit number), the call sign, and the vehicle type. This format provides a quick lookup to correlate all entity information. The vehicle table does not provide a listing of aggregated units, only individual entities (see figure 2).

The second file is the direct-fire file. Information includes the time of the hit, the firer, the target, the location of the firer and target, the ammunition used, the range, and the outcome. The information is provided for every direct-fire occurrence. In the example shown in figure 3, an entity (VTAB identifier or VTAB_ID of 1001) is hit with a U. S. M392A2 (a 105-mm

```
VTAB_ID 1013 PO_VEHICLE 100B51 VEHICLE_TYPE vehicle_USSR_BMP2
VTAB_ID 1006 PO_VEHICLE 100B52 VEHICLE_TYPE vehicle_USSR_BMP2
VTAB_ID 1014 PO_VEHICLE 100B22 VEHICLE_TYPE vehicle_USSR_T72M
VTAB_ID 1015 PO_VEHICLE 100B21 VEHICLE_TYPE vehicle_USSR_T72M
VTAB_ID 1018 PO_VEHICLE 100A82 VEHICLE_TYPE vehicle_USSR_T72M
VTAB_ID 1037 PO_VEHICLE 100A62 VEHICLE_TYPE vehicle_USSR_T80
VTAB_ID 1002 PO_VEHICLE 100A51 VEHICLE_TYPE vehicle_USSR_T80
VTAB_ID 1001 PO_VEHICLE 100A52 VEHICLE_TYPE vehicle_USSR_T80
VTAB_ID 1003 PO_VEHICLE 100A21 VEHICLE_TYPE vehicle_US_M1
VTAB_ID 1012 PO_VEHICLE 100A14 VEHICLE_TYPE vehicle_US_M1
```

Figure 2. Vehicle table sample.

```
Time Stamp 1122577497
Firer ID 1012
Target ID 1001
Firer Position:  X = 3689.51  Y = 45022.77  Z = 1116.82
Target Position:  X = 3966.29  Y = 42459.20  Z = 1149.10
Vehicle 1001: Hit with 1 "munition_US_M392A2" (0x48b80421)
  Comp DFDAM_EXPOSURE_TURRET, angle 41.01 deg Disp 4.085521 ft
  Kill Thermometer is: Pk: 1.00, Pmf: 0.60, Pf: 0.40, Pm: 0.30
  Pn:  0.30
r = 0.957620 kill_type = K
RANGE  2578.668050
```

Figure 3. Direct-fire event.

armor-piercing discarding sabot-tracer). The target (VTAB_ID of 1001) is a T-80 main battle tank and the firer (VTAB_ID of 1012) is an M1 main battle tank. This additional information can be obtained from the vehicle table. The result of this hit is a K or catastrophic kill. These tables provide invaluable information on what is happening during the simulation execution. However, not all damage is a result of a direct fire, so additional information needed to be obtained from OneSAF to handle indirect-fire events.

3. Methods/Procedures

3.1 Data Collection Modifications

The original KVS data collection capability collected data whenever the simulation entered a certain software routine. The routine was located in the statmon_init.c program in the libsrc/libstatmon directory. This provided the data as seen in figure 1. However, this methodology had two limitations. The first limitation was that the data were only collected when the simulation entered the specified routine, so data were only available at intermittent times. There was no guarantee that data would be collected on a regular basis. The second limitation was that the user had to turn on the status function in the OneSAF graphical user interface (GUI) to ensure that data was collected. Changes were made to overcome these two limitations.

To correct the first limitation, the code was modified in two separate libraries. The first change provided the ability to set the appropriate time increment to collect data. A user can specify the collection time frequency by entering a value, in seconds, in the file INCREMENT located in the OneSAF home directory. This file is read at simulation startup. If the file does not exist, the simulation will collect data every 60 s (default value). This change was made in the ent_tick.c file located in the <OneSAF-home>/libsrc/libentity directory. The second library changed was the <OneSAF-home>/libsrc/libstatmon. (Please see appendices A and B for code changes.) We added a new software routine called perpetual_collect_data in the stmn_init.c file. This routine allows for the collection of data at the default or user-defined interval. The collected data are placed in a file named with the unique timestamp for that simulation run with a *dc* extension and placed in a directory at the OneSAF home level entitled PERPETUAL. The format for the data is the same as shown in figure 1.

We overcame the second limitation by initiating changes in the ent_tick.c file. We modified the code so the data are collected without user intervention. The original data collection process required that the user turn on the *Unit Status* function in the OneSAF GUI when the simulation was executing. If the user did not select *Unit Status*, then no data would be collected. These modifications allow for constant data collection without relying on the user to initiate this functionality. One caveat must be applied: the user must monitor disk space on the system. The data collection files can grow quite large and could impede simulation execution if disk space is impacted.

While the changes allow for constant data collection, the original functionality is intact. If a user does select *Unit Status* the system will still create a file with the *dc* extension in the DATACOLL directory. It is possible that two files will be created, one in the DATACOLL directory and one in the PERPETUAL directory. The PERPETUAL file will always be created when the simulation is running. The DATACOLL file will be created if the user selects the *Unit Status* function. The two files will contain data collected at different time points in the simulation execution.

3.2 Indirect-Fire Modifications

The OneSAF code was modified to provide a real-time list of all indirect-fire events to a text file for further processing. A new file is created for each simulation execution and is uniquely named with a timestamp and an *if* extension (e.g., 1121772206if). The files for the collected data, the direct-fire, and vehicle table are also named with the same timestamp so all information from one simulation run can be correlated. Using the previous example, the data collection file would be 1121772206dc, the direct-fire file would be 1121772206df, and the vehicle table would be 1121772206vt. The indirect-fire file would be placed in the directory <OneSAF-home>/IFKVS.

To create the file that contains the indirect-fire information, new code was inserted into the indirect-fire library (OneSAF directory/libsrc/libifdam). (Please see appendix C for code changes.) The modified program is ifdam_tick.c. The program remains in a wait state and is always available to assess indirect-fire damageas the event occurs. When indirect-fire is detected, all entities in the local area are assessed for damage. OneSAF calculates all entities within the range of a detonation based on the ammunition and then assesses damage for all within-range entities. The event will print out one entry for every vehicle that can be impacted by the event.

Information that is provided for the indirect-fire event includes a time stamp when the event occurred, the vehicle that is being assessed, the munitions used, the location of the entity and the detonation, and the outcome. In the example shown in figure 4, three vehicles were within the range of influence from an indirect-fire event. All three entries are from the same event, as is evidenced by the identical time stamp and the same detonation location. The detonation was from a U.S. M712, a 155-mm Copperhead. Two vehicles sustained no damage, while one was firepower killed. From looking at the vehicle table entry all three vehicles are USSR T-72 main battle tanks.

```
============================================
Time Stamp 1121772206
Vehicle 1015 assessing IF damage with 1 "munition_US_M712"
Entity Location X = 6737.77 Y = 41301.00 Z = 987.29
U No Damage
Detonation Location X = 6571.00  Y = 41190.42 Z = 991.25
============================================
Time Stamp 1121772206
Vehicle 1018 assessing IF damage with 1 "munition_US_M712"
Entity Location X = 6571.37 Y = 41190.00 Z = 991.23
F Fire Kill
Detonation Location X = 6571.00  Y = 41190.42 Z = 991.25
============================================
Time Stamp 1121772206
Vehicle 1014 assessing IF damage with 1 "munition_US_M712"
Entity Location X = 6599.09 Y = 41328.70 Z = 990.82
U No Damage
Detonation Location X = 6571.00  Y = 41190.42 Z = 991.25
============================================
```

Figure 4. Sample from the indirect-fire file.

4. Results and Discussion

The addition of indirect-fire events to the KVS increases the amount of data that can be captured from simulation runs. Additionally, changes recently made to the data collection capability decrease the amount of user interaction necessary. This will allow multiple simulations to be run in a batched mode with little dependence on a user. This feature will support our data mining efforts as the number of simulations executed can increase. The indirect-fire capability will provide more data to explain changes in entity state that are the result of events other than direct fire.

With regards to our BTO thrust, the KVS enhancements will increase the accuracy of the data obtained. BTO provides a dynamic view of the battlespace that visualizes a power projection of force control over terrain area. BTO constantly monitors for updates on entity status and location (information from the *dc* file) and direct-fire events (from the *df* file). Prior to the enhancements, if an indirect-fire event occurred, a vehicle status may have changed. However, there would be no fire event reporting why the change occurred. BTO will now have the information to explain the change in status if an indirect-fire event occurs and causes damage to a battlespace entity.

The OneSAF modifications were tested on systems running with the Red Hat Enterprise 3 Linux operating system. We used version 2.0, service patch 4 of OTB, for the KVS enhancements.

5. Conclusions/Recommendations

KVS has been an invaluable addition to the functionality of OTB. The data obtained from using KVS benefit projects here at ARL, e.g., BTO and data mining. KVS is available to the OTB community. Mark Hickie, U.S. Air Force, used the KVS functionality to support his research at Draper Laboratory (*5*). The enhancements made to the data collection process and the ability to track indirect-fire events will allow for a more complete dataset to be obtained during OTB execution.

The original KVS code is incorporated into OTB v2.0. Any authorized user of OTB can access the functionality for data collection and the KVS. The enhancements discussed in this report will be sent to PEO-STRI at the next data call.

With increased intelligence gathering and sensors, the future will bring more information to the battle commander. However, commanders cannot be overloaded or critical information may be lost. Techniques and methodologies being developed by BDST, e.g., BTO and data mining, will allow information in the future to be distilled for the commander. The enhancements made to KVS allow more information to be gathered from a simulation and will then rely on data mining and other technologies to provide a concise picture of the battlespace for the commander. Technologies being developed today are to ensure battle decisiveness in the future.

6. References

1. O'May, J.; Hansen, C.; Heilman, E.; Kaste, R.; Neiderer, A. Battlespace Terrain Ownership: A New Situation Awareness Tool. *Proceedings of the 10th International Command and Control Research and Technology Symposium*, McLean, VA, 2005.

2. Bodt, B.; Heilman, E.; O'May, J. *Battle Command Metric Exploration in a Simulated Combat Environment*; ARL-TR-3429; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, February 2005.

3. Heilman, E.; O'May, J. *A OneSAF Data Collection Methodology for Experimentation*; ARL-TR-2663; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, February 2002.

4. Heilman, E.; O'May, J. *One Semi-Automated Forces (OneSAF) Killer/Victim Scoreboard (KVS) Capability*; ARL-TR-2829; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, September 2002.

5. Hickie, M. M. Behavioral Representation of Military Tactics for Single-Vehicle Autonomous Rotorcraft via Statecharts. Master of Science Thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2005.

INTENTIONALLY LEFT BLANK.

Appendix A. Changes in OneSAF's libsrc/libstatmon Directory

This appendix appears in its original form, without editorial change.

Additions to libsrc/libstatmon/libstatmon.h:

Additions to libsrc/libstatmon/libstatmon.h:

```
/* const_value_to_name:
 *
 * Translate a constant to its name (returned as a libreader symbol).
 * namespace should be a libreader symbol.  If this fails, it returns
 * the value of const_error.
 */
extern char                    *const_value_to_name (
    const char *namespac,
    int32 value);

/* reader_get_symbol:
 *
 * Looks up a string in the symbol table and returns the equivalent
 * symbol.  Note that this will add the string to the symbol table
 * if it isn't already there.
 */
extern char                    *reader_get_symbol (
    const char *s);


/* pm_type_symbol:
 *
 * Generates the libreader symbol which corresponds to the passed
 * objectType and methodology.
 */
extern char                    *pm_type_symbol (
    ObjectType object_type,
    SAFMethodology methodology);

/* KVS_monitor_status
 *
 * ARL devloped routine to handle fuel and ammo information
 *
 */
extern void KVS_monitor_status (
    PO_DB_ENTRY * unit_entry,
    char *message);
```

Additions to libsrc/libstatmon/libstmn_local.h:

```
extern PO_DATABASE *joegh_po_db;

/* perpetual_status
 *
 * ARL developed routine for vehicle sensing
 *
 */
extern void perpetual_status (
    int32       vehicle_id,
    char description[]);
```

Additional subroutine added to libsrc/libstatmon/stmn_init.c:

```
/*
 * Subroutine to always print out status information to a file.
 * Information used for simulation data collection.
 */
void perpetual_collect_data ()
{
    int32                           i;
    struct timeval                  time_of_day;
    static int32                    new_arl_time;
    static int32                    count1 = 0;
    int32                           time_in_secs;
    PO_DB_ENTRY                    *entry;
    FILE                           *temp_file;
    char                            message[20480];


   /* extern PO_DATABASE *joegh_po_db; */
    /*
     * Creates a file name using a UNIX timestamp
     * with appended dc within the DATACOLL
     * directory.
     */


    bzero(message, 20480);

    if (count1 == 0)
    {
        fname1[0] = '\0';
        sprintf (fname1,
            "%s%d%s", "../../PERPETUAL/", basic_retrieve_arl_time (),
            "dc");
    }

    /*
     * Set this flag to ensure file is created only on the first code
     * execution per OneSAF run.
     */
    count1++;
    gettimeofday (&time_of_day, (struct timezone *) NULL);
    time_in_secs = time_of_day.tv_sec;

    /*
     * Code to open file and insert current timestamp information.
     */
    temp_file = fopen (fname1, "a");
    fprintf (temp_file,
        "Current Count %d at Time: %d \n\n", count1, time_in_secs);

    /*
```

```
     * This loop inserts the following information for each entity in
the
     * simulation: Object ID, location in x, y, and z, appearance, and
other
     * logistical information.
     */

    for (entry =
        joegh_po_db->
        object_classes[PO_OBJECT_CLASS_INDEX (objectClassUnit)]; entry;
        entry = entry->next_class)
    {
        UnitClass                       *unit = &PO_UNIT_DATA (entry);
        PO_DB_ENTRY                     *task_entry;


        if ( strcmp((char*)unit->marking.text, "\0") != 0 )
        {
        task_entry = taskfr_get_background_task (joegh_po_db,
            entry, SM_UEnemy);


        fprintf (temp_file, "-----------------------------------\n");
        fprintf (temp_file, "MARKING: %s\n", unit->marking.text);
        fprintf(temp_file, "VEHICLE: %s\n",
             pm_type_symbol (unit->objectType, unit->methodology));
        fprintf(temp_file, "VTAB:  %d\n", unit->simulationID.vehicle);
        fprintf (temp_file, "X = %.2f  Y = %.2f   Z = %.2f  CELL =
            %d\n", unit->location[X], unit->location[Y],
            unit->location[Z], (int32) unit->location[CELL3D]);
        KVS_monitor_status(entry,message);
        fprintf (temp_file, "%s\n", message);
        bzero(message, 20480);
        perpetual_status( unit->simulationID.vehicle, message);
        fprintf(temp_file, "%s\n", message);
        fprintf (temp_file, "-----------------------------------\n");
        }
    }

    fprintf (temp_file,
             "============================================\n");
    fflush (temp_file);
    fclose (temp_file);
}
```

Appendix B. Changes in OneSAF's libsrc/libentity Directory

This appendix appears in its original form, without editorial change.

Changes to file libsrc/libentity/ent_tick.c inside the ent_tick function call:

```
ent_tick ()

    struct timeval                time_of_day;
    int32                         new_tick_time = 0;
    static int32                  time_tick_interval = 0;
    static int32                  tick_count = 0;
    int32                         time_in_secs;
    static int32                  increment = 60;
    FILE                          *increment_file;
    static int                    increment_flag = 0;


    /* Only check for increment file once.
     * The increment will default to 60 seconds unless
     * there is another increment file. The increment
     * file should be in seconds.   */
    if ( increment_flag == 0 )
    {
       increment_file = fopen ( "../../INCREMENT", "r");

       if ( increment_file )
          fscanf ( increment_file, "%d", &increment);

       increment_flag = 1;
     }

    gettimeofday (&time_of_day, (struct timezone *) NULL);
    new_tick_time = time_of_day.tv_sec;

    if ( tick_count == 0 )
    {
       perpetual_collect_data();
       tick_count++;
       time_tick_interval = new_tick_time + increment;
    }
    else if ( new_tick_time >= time_tick_interval )
    {
       perpetual_collect_data();
       time_tick_interval = new_tick_time + increment;
    }
```

Appendix C. Changes in OneSAF's libsrc/libifdam Directory

This appendix appears in its original form, without editorial change.

Additions to libsrc/libifdam/libifdam_local.h:

```
/* Flag to ensure a single file name creation on any simulation run. */
extern int32                    if_data_file_created;

/* Placeholder for the ARL data file name (with extention) */
extern char                     ifname[1024];
```

Additions to libsrc/libifdam/ifdam_tick.c:

```
/* JO 14 Dec 2004 */
#include <assert.h>
#include <sys/time.h>
/* JO 14 Dec 2004 */


static void modify_ic_probability ()

    FILE                           *temp_file;
    char                            ifname[1024];


        temp_file = fopen ( ifname, "a");
        fprintf(temp_file, "Detonation Location X = %.2f Y = %.2f Z =
                %.2f\n", location[X], location[Y], location[Z]);
        fprintf(temp_file, "Entity Location X = %.2f Y =  %.2f Z =
                %.2f]\n", vehicle_location[X], vehicle_location[Y],
                vehicle_location[Z]);
        fprintf(temp_file, "p(k) %f p(mf) %f p(f) %f p(m) %f p(n)
                %f\n", calc_prob->k, calc_prob->mf,
               calc_prob->f, calc_prob->m, calc_prob->n);
        fflush(temp_file);
        fclose(temp_file);

}


static void ifdam_lookup_probabilities ()

    FILE                   *temp_file;
    char                            ifname[1024];



    sprintf (ifname, "%s%d%s", "../../IFKVS/",
             basic_retrieve_arl_time (), "if");
    temp_file = fopen ( ifname, "a");
    fprintf(temp_file, "Entity Location X = %.2f Y = %.2f Z = %.2f\n",
            vehicle_location[X],vehicle_location[Y],
            vehicle_location[Z]);
    fprintf(temp_file, "Shooter ID %d\n", shooter_id);
    fflush(temp_file);
    fclose(temp_file);



static void ifdam_take_damage ()
{
    /* JO 21 Jun 2005 */
    char                            ifname[1024];
    FILE                           *temp_file;
    /* JO 21 Jun 2005 */



    if (!is_ic)
    {
        if (r <= therm_prob.n)
        {
            /*
             * no kill or damage from this round
             */

           /* JO 1 Jun 2005 */
           sprintf (ifname, "%s%d%s", "../../IFKVS/",
                    basic_retrieve_arl_time (), "if");
           temp_file = fopen ( ifname, "a");
           fprintf(temp_file, "U No Damage \n");
           fflush(temp_file);
           fclose(temp_file);
           /* JO 1 Jun 2005 */
        }
        else if (udam_handle_kill (vehicle_id, TRUE))
        {

        }
        else if (r <= therm_prob.m)
        {
           /* JO 1 Jun 2005 */
           sprintf (ifname, "%s%d%s", "../../IFKVS/",
                    basic_retrieve_arl_time (), "if");
           temp_file = fopen ( ifname, "a");
           fprintf(temp_file, "M Mobility Kill \n");
           fflush(temp_file);
           fclose(temp_file);
           /* JO 1 Jun 2005 */


        }
        else if (r <= therm_prob.f)
        {
            DELETED CODE
           /* JO 1 Jun 2005 */
           sprintf (ifname, "%s%d%s", "../../IFKVS/",
                    basic_retrieve_arl_time (), "if");
           temp_file = fopen ( ifname, "a");
           fprintf(temp_file, "F Fire Kill \n");
           fflush(temp_file);
```

```
         fclose(temp_file);
         /* JO 1 Jun 2005 */

      }
      else if (r <= therm_prob.mf)
      {
         /* JO 1 Jun 2005 */
         sprintf (ifname, "%s%d%s", "../../IFKVS/",
                  basic_retrieve_arl_time (), "if");
         temp_file = fopen ( ifname, "a");
         fprintf(temp_file, "K Catastrophic Kill \n");
         fflush(temp_file);
         fclose(temp_file);
         /* JO 1 Jun 2005 */

          }
          else
          {
          DELETED CODE

         /* JO 1 Jun 2005 */
         sprintf (ifname, "%s%d%s", "../../IFKVS/",
                  basic_retrieve_arl_time (), "if");
         temp_file = fopen ( ifname, "a");
         fprintf(temp_file, "MF Mobility & Fire Kill \n");
         fflush(temp_file);
         fclose(temp_file);
         /* JO 1 Jun 2005 */

      }
      else
      {
         /* JO 1 Jun 2005 */
         sprintf (ifname, "%s%d%s", "../../IFKVS/",
                  basic_retrieve_arl_time (), "if");
         temp_file = fopen ( ifname, "a");
         fprintf(temp_file, "K Catastrophic Kill \n");
         fflush(temp_file);
         fclose(temp_file);
         /* JO 1 Jun 2005 */

  }
  else
  {

      switch (ic_vunerable_factor)
      {
          case KILL:

              if (r <= therm_prob.n)
              {
                  /* JO 1 Jun 2005 */
                sprintf (ifname, "%s%d%s", "../../IFKVS/",
                         basic_retrieve_arl_time (), "if");
                temp_file = fopen ( ifname, "a");
                fprintf(temp_file, "U No Damage \n");
                fflush(temp_file);
                fclose(temp_file);

              }
              else
              {
                  /* JO 1 Jun 2005 */
                  sprintf (ifname, "%s%d%s", "../../IFKVS/",
                           basic_retrieve_arl_time (), "if");
                  temp_file = fopen ( ifname, "a");
                  fprintf(temp_file, "K Catastrophic Kill \n");
                  fflush(temp_file);
                  fclose(temp_file);
                  /* JO 1 Jun 2005 */

          default:
              if (r <= therm_prob.n)
              {
                  /* JO 1 Jun 2005 */
                  sprintf (ifname, "%s%d%s", "../../IFKVS/",
                           basic_retrieve_arl_time (), "if");
                  temp_file = fopen ( ifname, "a");
                  fprintf(temp_file, "U No Damage \n");
                  fflush(temp_file);
                  fclose(temp_file);
                  /* JO 1 Jun 2005 */
              }
              else if (r <= therm_prob.m)
              {
              /* JO 1 Jun 2005 */
              sprintf (ifname, "%s%d%s", "../../IFKVS/",
                       basic_retrieve_arl_time (), "if");
              temp_file = fopen ( ifname, "a");
              fprintf(temp_file, "M Mobility Kill \n");
              fflush(temp_file);
              fclose(temp_file);
              /* JO 1 Jun 2005 */
              }
              else if (r <= therm_prob.f)
              {
                 /* JO 1 Jun 2005 */
                 sprintf (ifname, "%s%d%s", "../../IFKVS/",
                          basic_retrieve_arl_time (), "if");
                 temp_file = fopen ( ifname, "a");
                 fprintf(temp_file, "F Fire Kill \n");
                 fflush(temp_file);
                 fclose(temp_file);
                 /* JO 1 Jun 2005 */

              }
              else if (r <= therm_prob.mf)
              {
                  if (VTAB_TYPES_MATCH (vtab_vehicle_type
                      (vehicle_id), VTAB_STRUCTURE))
                  {
                      /* JO 1 Jun 2005 */

                      sprintf (ifname, "%s%d%s", "../../IFKVS/",
                                  basic_retrieve_arl_time (), "if");
                         temp_file = fopen ( ifname, "a");
                         fprintf(temp_file, "K Catastrophic Kill\n");
                         fflush(temp_file);
                         fclose(temp_file);
                         /* JO 1 Jun 2005 */

                     }
                     else
                     {
                         /* JO 1 Jun 2005 */
                         sprintf (ifname, "%s%d%s", "../../IFKVS/",
                                  basic_retrieve_arl_time (), "if");
                         temp_file = fopen ( ifname, "a");
                         fprintf(temp_file, "MF Mobility & Fire Kill
                                 \n");
                         fflush(temp_file);
                         fclose(temp_file);
                         /* JO 1 Jun 2005 */

                 }
                 else
                 {
                     /* JO 1 Jun 2005 */
                     sprintf (ifname, "%s%d%s", "../../IFKVS/",
                              basic_retrieve_arl_time (), "if");
                     temp_file = fopen ( ifname, "a");
                     fprintf(temp_file, "K Catastrophic Kill\n");
                     fflush(temp_file);
                     fclose(temp_file);
                     /* JO 1 Jun 2005 */


    }

}



static void ifdam_received_pdu ()
{

/* JO 13 Dec 2004 */
    FILE                *temp_file;     /* Placeholder to contain file
                                         * name for the fopen UNIX
                                         * function */

    struct timeval      tick_time;      /* UNIX structure for results
                                         * of time function call. */

    int32               new_tick_time;  /* Placeholder for the
                                         * current timestamp. */
/* JO 13 Dec 2004 */



/*
```

```
      * Set up code for filename
      */
     if (!if_data_file_created)
     {
         ifname[0] = '\0';
         sprintf (ifname, "%s%d%s", "../../IFKVS/",
                  basic_retrieve_arl_time (), "if");
         if_data_file_created = TRUE;
     }


/* JO 13 Dec 2004 */
     /*
      * Open data file and appending various entity information.
      */
     temp_file = fopen (ifname, "a");
     if (!temp_file)
     {
         fprintf (stderr,
             "libdfdam dfdam_received_detonate, failed to open file
              %s.\n", ifname);
         assert (temp_file);
         return;
     }
/* JO 13 Dec 2004 */



             /* JO 13 Dec 2004 */
             fprintf (temp_file,
                      "============================================\n");

             gettimeofday (&tick_time, (struct timezone *) NULL);
             new_tick_time = tick_time.tv_sec;
             fprintf(temp_file, "Time Stamp %d\n", new_tick_time);
             fprintf (temp_file, ("Vehicle %d assessing IF damage with
                      %d \"%s\" \n"),
             vehicle_id,
             quantity,
             const_value_to_name (reader_get_symbol ("munition"),
                (int32) projectile));

             fflush(temp_file);
             fclose(temp_file);
             /* JO 13 Dec 2004 */



         {

            /* JO 11 Jul 05 */
             temp_file = fopen (ifname, "a");
             fprintf(temp_file,"Detonation Location X = %.2f  Y = %.2f
                     Z = %.2f\n", location_gcs[X],
                     location_gcs[Y], location_gcs[Z]);
             fflush(temp_file);
             fclose(temp_file);
```

```
              /* JO 11 Jul 05 */
          }
      }

  }
```

List of Symbols, Abbreviations, and Acronyms

ARL	U. S. Army Research Laboratory
BDST	Battlespace Decision Support Team
BTO	Battlespace Terrain Ownership
GUI	Graphical User Interface
KVS	Killer/Victim Scoreboard
OneSAF	One Semi-Automated Forces
OTB	OneSAF Testbed Baseline
PEO-STRI	Program Executive Office for Simulation, Training, and Instrumentation Command

NO. OF COPIES	ORGANIZATION
1 (PDF ONLY)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN AUSTIN TX 78759-5316
1	DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES	ORGANIZATION
1	PM ONESAF LTC J SURDU PEO STRI 12350 RESEARCH PKWY ORLANDO FL 32826-3276
	ABERDEEN PROVING GROUND
12	DIR USARL AMSRD ARL CI CT CHIEF C HANSEN E HEILMAN R KASTE A NEIDERER J OMAY (3 CPS) M THOMAS AMSRD ARL WM BF P BUTLER C PATTERSON G SAUERBORN

INTENTIONALLY LEFT BLANK.